METODE PENCARIAN DAN TEMU-KEMBALI NAMA BERDASARKAN KESAMAAN FONETIK

Dewi Primasari
Program Studi Teknik Informatika, Fakultas Teknik
Unversitas Ibn Khaldun Bogor
Jln. K.H Sholeh Iskandar Km. 2 Bogor
e-mail: dewiprimasari@yahoo.com

***Abstrak* — Sistem temu-kembali informasi membandingkan query pengguna pada dokumen-dokumen yang disimpan dalam berkas basis data. Beberapa pendekatan pembentukan turunan query telah dikembangkan untuk mengatasi ketidakjelasan akar kata yang akan dicari, salah satunya adalah dengan ukuran kesamaan fonetik. Ada beberapa algoritme dengan menggunakan pendekatan ini, yaitu Soundex dan Phonix yang dikembangkan berdasarkan kesamaan bunyi dalam bahasa Inggris.**

Pada penelitian ini akan dibuktikan bahwa algoritme Phonix4 dan PhonixE memiliki kinerja yang lebih baik bila dibandingkan dengan algoritme Phonix8 dan Soundex. Hal ini terjadi karena kesederhanaan algoritme Phonix4 dan PhonixE, dimana kode filtering yang dibentuk tidak terlalu panjang, dan penggolongan konsonan yang tidak terlalu sedikit. Akibatnya nilai precision menjadi tinggi, sedangkan tingkat recall-nya rendah. Selain kode yang biasa dipakai pada metode lainnya, PhonixE juga mempunyai kode untuk bunyi akhir sebanyak empat karakter. Tingkat recall yang dapat dipakai pada system temu-kembali informasi dengan Soundex dan Phonix8 adalah 0.2, sedangkan pada metode Phonix4 dan PhonixE adalah 0.1.

Algoritme Soundex dan Phonix walaupun dikembangkan untuk bahasa Inggris, ternyata dapat berkerja dengan baik pada bahasa Indonesia. Hal tersebut karena bahasa Inggris dan Indonesia menggolongkan konsonan-konsonannya denan keistimewaan-keistimewaan yang sama.

***Kata kunci*: pencarian, temu-kembali, fonetik, Soundex, Phonix**

I. PENDAHULUAN

A. Latar Belakang

Sistem temu-kembali upada dokumen berbahasa Indonesia sering sekali mengalami kesulitan karena adanya berbagai ejaan yang perah berlaku di Indonesia, Penemu-kembalian dilakukan dengan membandingkan query pengguna pada dokumen-dokumen yang disimpan dalam basis data (Pfeifer et al., 1996). Istilah dokumen (document) pada penelitian ini mempunyai arti yang berbeda, yaitu sebuah obyek data, yang biasanya berbentuk teks walaupun dapat juga terdiri dari berbagai tipe data, seperti foto, grafik, dan sebagainya (Frakes & Beza-Yates, 1992).

Sistem temu-kembali informasi (information retrieval) secara otomatis pada awalnya dikembangkan untuk membantu mengatur literatur ilmu pengetahuan yang jumlahnya sangat banyak. Sistem temu-kembali informasi modern yang telah ada umumnya difokuskan pada masalah-masalah temu-kembali informasi dengan query-query berkonteks bahasa natural. Beberapa pendekatan telah dibuat, seperti dengan menggunakan wildcard yang terperinci pada query untuk menemukan turunan atau kata yang mirip. Ukuran kesamaan non-linguistik dapat dikelompokan menjadi 3 kategori besar yang berbeda yaitu (1) kesamaan string, (2) kesamaan yang berhubungan dengan kesalahan pengetikan, dan (3) kesamaan fonetik. Pada penelitian ini akan dilakukan pencarian nama dengan metode kategori ketiga, yaitu dengan menggunakan algoritme Phonix.

Algoritme Phonix adalah algoritme yang dibuat dengan melihat kesamaan bunyi bahasa. Algoritme ini dikembangkan untuk bahasa Inggris, yang belum tentu cocok bila diterapkan pada bahasa lain, misalnya bahasa Indonesia yang pengucapannya berbeda. Maka pada penelitian ini akan ditelaah apakah algoritme tersebut dapat bekerja dengan baik pada bahasa Indonesia.

Pada dasarnya proses temu-kembali informasi dilakukan berdasarkan ukuran kesesuaian antara query dengan kata-kata yang terdapat dalam dokumen. Yang menjadi masalah adalah pada sistem temu-kembali informasi, jumlah dokumen yang relevan kadang-kadang terlalu sedikit atau terlalu banyak (Adisantoso, 1977). Untuk itu perlu diketahui berapa jumlah dokumen yang ditemu-kembalikan agar cukup optimal.

Sebenarnya penelitian untuk membandingkan kinerja beberapa metode terkenal dari ketiga kategori ukuran kesamaan di atas telah dilakukan oleh Ulrich Pfeifer, Thomas Poersh, dan Norbert Fuhr pada tahun 1996. Pada penelitian tersebut basis data yang digunakan adalah nama-nama orang Eropa atau Amerika. Sedangkan pada penelitian dengan menggunakan basis data dengan nama-nama Indonesia yang umum telah dilakukan oleh Dewi Primasari (1997). Penelitian lanjutan ini akan menggunakan basis data yang sama, tetapi dengan modifikasi algoritme Phonix.

B. Tujuan

Tujuan dari penelitian ini adalah sebagai berikut :

1. Menelaah dan membandingkan efektifitas sistem temu-kembali antara algoritme varian Phonix yang telah dimodifikasi agar agar tidak terpengaruh ejaan lama.
2. Menganalisis kinerja masing-masing algoritme yang telah diperbaiki terhadap basis data yang berisi nama-nama Indonesia
3. Menentukan jumlah dokumen (tingkat recall) yang ditemu-kembalikan untuk setiap metode agar cukup optimal.

II. TINJAUAN PUSTAKA

Sistem temu kembali informasi secara otomatis mengelola dokumen-dokumen yang terorganisasi dalam

record pada berkas (*file*) dan mengelola permintaan (*request*) informasi, kemudian mengembalikannya dalam berkas tertentu sebagai tanggapan terhadap permintaan tersebut. Penemu-kembalian dokumen-dokumen tergantung pada ukuran kesamaan antara dokumen dan *query* yang diukur dengan membandingkan nilai berapa atribut.

Sistem temu-kembali informasi harus dibuat untuk mendukung operasi-operasi dasar, yaitu operasi pemasukan dokumen ke dalam basis data, operasi penambahan dokumen, operasi penghapusan, operasi pencarian dokumen dengan cara tertentu, dan operasi penampilan pada layar.

A. Organisasi Data

1. Struktur Data

Struktur data dalam temu-kembali informasi sedikit banyak akan mempengaruhi efisiensi kerja sistem temu-kembali informasi tersebut, terutama dalam proses pencarian (Frakes & Baeza-Yates, 1992). Oleh karena itu tipe struktur data temu-kembali informasi ini harus dipilih dengan hati- hati. Ada empat bentuk dasar untuk mengorganisasikan data, yaitu (1) larik (*array*), (2) *search tree* , (3) *digital tree*, dan (4) *hash.* Tipe larik adalah tipe terstruktur yang mempunyai komponen dalam jumlah yang tetap dan setiap komponen mempunyai tipe data yang sama (Santoso, 1973).

2. Pengurutan

Pengurutan data (sorting) secara umum didefinisikan sebagai suatu proses untuk menyusun kembali himpunan objek menggunakan aturan tertentu (Santoso, 1993). Secara umum terdapat dua jenis pengurutan data, yaitu (1) pengurutan secara menaik (ascending), yaitu dari data yang nilainya paling kecil sampai paling besar dan (2) pengurutan secara menurun (descending), yaitu dari data yang nilainya paling besar sampai paling kecil. Tujuan pengurutan data adalah untuk mempermudah pencaraian data.

Pemilihan algoritme pengurutan sangat ditentukan oleh struktur data yang digunakan. Dengan alasan ini, maka metode pengurutan dapat diklasifikasikan menjadi dua kategori, yaitu pengurutan larik dan pengurutan senarai.

Metode-metode pengurutan larik sangat memperhatikan dan mempertimbangkan aspek efisiensi waktu dan kapasitas memori. Secara umum metode quick sort lebih efisien daripada metode-metode lainnya (Stubbs & Webre, 1984), algoritmenya adalah sebagai berikut :

1) Baca larik yang akan diurutkan misalnya r.
2) Inisialisasi kiri = 1 dan kanan = jumlah record yang ada.
3) Kerjakan langkah 4 sampai 5 bila kiri < kanan.
4) Inisialisasi j = kiri dan k = kanan + 1
5) Kerjakan langkah 6 hingga 8 sampai j > k.
6) Tambah nilai j dengan 1 sampai r[j[] >= r[kiri].
7) Kurangi nilai k dengan 1 sampai r[k] <= r[kiri]
8) Jika j < k, maka tukarkan posisi r[j] dengan posisi r[k].
9) Tukarkan posisi r[kiri] dengan r[k].
10) Kerjakan langkah 3 untuk kiri = 1 dan kanan = k - 1
11) Kerjakan langkah 3 untuk kiri = k + 1 dan kanan = jumlah record yang ada.

3. Pencarian

Metode yang paling sederhana dari sejumlah metode pencarian adalah metode pencarian berurutan (sequential searching). Cara kerja metode ini, mula-mula adalah melakukan pembacaaan data yang akan dicari, data yang dicari dibandingkan satu persatu dengan data pertama sampai data terakhir secara berurutan, hingga data tersebut ditemukan atau tidak ditemukan, maka proses pencarian dapat langsung dihentikan. Jika data yang dicari belum ditemukan, maka pencarian terus dilakukan sampai seluruh data dibandingkan. Dalam kasus terburuk untuk vector dengan N buah elemen harus dilakukan pencarian sebanyak N kali pula (Santoso, 1993).

Algoritme pencarian berurutan adalah sebagai berikut :

1) Baca larik yang diketahui, misalnya larik A dengan N element.
2) Baca data yang dicari, misalnya diberi nama data.
3) Inisialisasi ada = false dan posisi = 0
4) Dalam proses pencarian, untuk i = 1 sampai N dikerjakan langkah 5.
5) Tes apakah Data = A[i].
6) Jika ya (berarti data telah ditemukan), tentukan posisi = i, ada = true dan i = N.

Jika dianggap tidak terjadi penambahan dan penghapusan elemen, berarti pencarian dilakukan pada tabel yang berukuran tetap, yaitu N. maka banyaknya perbandingan bergantung erhadap letak elemen yang dicari, jika elemen yang dicari terletak sebagai elemen pertama, berarti perbandingan cukup dilakukan sebanyak satu kali. Sebaliknya jika elemen yang dicari terletak sebagai elemen terakhir, maka perlu dilakukan pembandingan sebanyak N kali. Secara singkat untuk pencarian yang berhasil rata-rata diperlukan pembandingan sebanyak (N+1)/2 kali dan pencarian yang tidak berhasil memerlukan N kali pembandingan. Notasi yang sering digunakan adalah O(N). Perhitungan ini dilakukan dengan asumsi bahwa dalam berkas tersebut tidak terdapat data yang sama. Untuk menghindari pencarian secara berurutan, digunakan pencarian secara acak.

B. Algoritme

Algoritme temu-kembali informasi dapat diklasifikasikan menjadi tiga buah tipe, walaupun sebenarnya sulit untuk menentukan batas yang jelas pada masing-masing tipe. Tiga tipe tersebut adalah algoritme temu-kembali, algoritme filtering, dan algoritme pengindeksan (Frakes & Baeza – Yates, 1992).

1. Algoritme Temu-Kembali

Algoritme temu-kembali adalah algoritme untuk mengekstrak informasi dari basis data tekstual. Algoritme ini dapat dibedakan menjadi dua buah tipe menurut banyak memori tambahan yang diperlukan, yaitu (1) algoritme scan sequential text dan (2) algoritme teks yang diindeks (Frakes & Baeza -Yates, 1992).

Algoritme pertama (algoritme scan sequential text) membutuhkan memori tambahan yang dalam worst case-nya merupakan fungsi dari ukuran basis datanya. Di pihak lain, (running time-nya proporsional dengan ukuran teks, misalnya dalam pencarian string (string searching). Jenis algoritme pencarian string sangat banyak, diantarnya adalah algoritme Naïve, algoritme Knuth-Morris-Pratt, dan algoritme Boyer-Moore-Horspool (Salton, 1989).

Algoritme kedua (algoritme teks yang diindeks) melakukan Pengindeksan terhadap teks dan dapat digunakan untuk meningkatkan kecepatan pencarian. Ukuran indeks

biasanya proporsional dengan ukumn basis data dan waktu pencarian sublinear terhadap ukuran teks, contohnya inverted file. Konsep dan tipe indeks inverted file adalah sebagai berikut : diasumsikan terdapat sekmnpulan dokumen, dimana setiap dokumen dicirikan oleh sekumpulan kata kunci atau atribut dengan bobot tambahan yang cocok dan dihubumgkan kepada setiap kata kunci. Jadi inverted file adalah daftar yang diurutkan oleh kata ktmci, di mana setiap kata kunci mcmptmyai hubungan dengan suatu dokumen yang mcngandung kata kunci. Fungsi inverted file adalah untuk mcmperbaiki efisiensi pencarian dengan beberapa ukuran jarak yang biasanya dipcrlukan untuk berkas teks yang besar. Ada dua algoritme pencarian untuk inverted file ini, yaitu algoritme pencarian kata kunci dan algoritme possible search. Algoritme pencarian kata kunci mengembalikan sebuah indeks sebagai hasil pencariannya, sedangkan algoritme possible search mengcmbalikan sekumpulan dokumen sebagai hasil pencariannya. Beberapa struktur yang dapat diimplementasikan pada inverted file, misalnya larik yang diurutkan (sorted array), B-Tree, dan Tries. Pada metode ini inverted file diimplementasikan sebagai struktur larik yang diurutkan dengan metode tertentu dan menyimpan daftar kata kumci dalam larik tersebut, tcrmasuk jumlah dokumen yang dihubungkan dengan setiap kata kunci dan sebuah penghubung untuk dokumen yang mcngandumg kata kunci. Kerugian mcnggunakan metode ini adalah rendahnya efisiensi kctika melakukan pemeliharaan indeks. Sedangkan keuntungannya adalah kcmudahan dalam pengimplementasiannya dan kecepatannya dalam pencarian. Pada prinsipnya, penemu-kembalian dokumen-dokumen yang disimpan untuk menjawab permintaan harus berdasarkan penentuan ukuran kesamaan antara query dan item yang disimpan dan mcngambil item-item yang dapat dibuktikan cukup mirip dengan query. Padahal dokumen-dokumen yang disimpan dan informasi yang diminta dapat tidak terstruktur dan keputusan penemu-kembaliannya tergantung pada teks yang berhubungan. Dalam keadaan ini, pembandingan secara langsung tidak tepat, sehingga diperlukan langkah-langkah antara untuk mentranformasi pcrmintaan kc dalam pcmyataan query formal dan dokumen ke dalam representasi indeks sebelum pembandingan dilakukan.

Dalam lingkungan operasional, dokumen-dokumen yang disimpan direpresentasikan oleh himpunan istilah (term) yang diindeks dan disebut vektor istilah (term vector). Kadang-kadang istilah tidak diberi bobot, walaupun pada beberapa situasi istilah diberi bobot untuk merefleksikan urutan relatif tingkat pentingnya. Operasi pencarian harus efektif dalam mengidentifikasi dokumen- dokumen yang disimpan. Berbagai strategi pencarian dan struktur data dapat digunakan untuk menghindari pencarian secara berurutan. Solusi yang biasa digunakan adalah dengan membentuk kepadatan indeks yang terpisah pada setiap istilah atau setiap nilai atritmt pada sistem. Maka untuk setiap istilah, indeks yang terpisah dikonstruksi untuk menyimpan pengenal (identifier) dokumen untuk setiap dokumen yang diidentifikasi.

Kumpulan indeks untuk setiap istilah dan nilai atribut tersebut disebut inverted index atau inverted file. Dengan inverted index ini, kumpulan dokumen yang berhubungan dengan formulasi query dengan mudah dapat ditentukan. Pengenal untuk semua item yang ditemu-kembalikan dapat diperoleh dengan mcngekstrak daftar dokumen pengenal inverted index yang bcrhubungan dengan masing-masing query istilah dan mcngkombinasikannya secara benar. Operasi inverted index mcmberikan jawaban query berdasarkan proses list merging yang menggabungkan dua atau lebih baris dari larik dokumen inverted istilah yang hasilnya berupa sebuah daftar pengenal dokumen.

2. Algoritme Filtering

Algoritme filtering adalah algoritme yang melakukan pemrosesan atau pemilteran, dimana masukan yang berupa teks diproses atau difilter dengan teks versi baru sebagai keluarannya. Ukuran kesamaan non-linguistik untuk algoritme filtering ini dapat dibedakan mcnjadi tiga kategori, yaitu (1) kesamaan string, (2) kesamaan yang berhubungan dengan kesalahan pengetikan, dan (3) kesamaan fonetik (Pfeifer et al., 1996). Untuk kategori pertama, kesamaan string, metode n-grams sangat sering dipakai. Sedangkan untuk kategori kedua, kesamaan string yang berhubungan dengan kesalahan pengetikan, metode Damerau-Levenstein-Metric digunakan dengan menghitung penyisipan, penghapusan, substitusi, dan transposisi yang dibutuhkan untuk mentransformasikan satu string ke string yang lain,

Dalam dua kelas ini pembandingan kata·kata dilakukan tanpa memperdulikan jenis bahasa yang digunakan. Sedangkan untuk jenis kategori ketiga, yaitu ukuran kesamaan fonetik, maka ukuran kesamaan fonatiknya tergantumg pada bahasanya. Algoritme Phonix adalah metode yang terkenal untuk membandingkan kata dengan memperhatikan kesamaan fonetik ini.

Kesamaan Fonetik. Sebagai contoh, asumsikan pembicara berbahasa lnggris menginstruksikan suatu perintah secara lisan kepada dua orang untuk mencari pengarang Jarman bemama BAYER. Seseorang yang berbahasa Inggris dapat melihat bahwa kata BUYER atau kata BYER dilafalkan sama dengan kata yang didengar.

Algoritme Phonix menghitung kode fonetik untuk setiap nama yang diberikan. Nama-nama yang berbagi dengan kode yang sama diasumsikan mirip (Pfeifer et al., 1996), Algoritme untuk menghitung kode Phonix menggunakan aturan—aturan penggantian yang lebih rumit atau teliti.
Karena algoritme tersebut dikembangkan untuk bahasa Inggris, maka harus dilakukan modifikasi bila akan diterapkan pada bahasa lain. Hal ini dapat dilakukan dengan mengadaptasi kelas-kelas huruf atau aturan·aturan pengganti.

Tabel 1. Bilangan pengganti huruf pada kode Phonix

Phonix						
B	F				→	1
C	G	J	K	Q	→	2
D	T				→	3
L					→	4
M	N				→	5
R					→	6
F	V				→	7
S	X	Z			→	8

Pada prinsipnya, cara kerja Phonix adalah sebagai berikut:

1) Jika huruf pertama adalah huruf hidup atau konsonan Y,

maka ganti huruf pertama tersebut dengan huruf V.
2) Buang bunyi akhir (ending sound) dari kata. Secara kasar bunyi akhir adalah bagian sesudah huruf vokal terakhir atau huruf Y.
3) Buang semua huruf hidup, konsonan H,W, dan Y, dan semua huruf sama yang berurutan.
4) Buat kode Phonix dari kata tersebut tanpa bunyi akhir dengan mengganti semua huruf yang tersisa dengan nilai numerik seperti pada Tabel 1, kecuali huruf pertama. Panjang maksimum kode Phonix dibatasi sampai delapan karakter.
5) Buat kode Phonix dari bunyi akhir dengan mengganti setiap huruf dengan nilai numerik.

Berdasarkan penelitian awal Primasari (1997), tabel Phonix (Tabel 2) tetap dapat digunakan, hanya algoritmenya saja yang mendapat perubahan.

Tabel 2. Bilangan pengganti huruf pada modifikasi kode Phonix

Modifikasi Phonix						
B	P				→	1
C	G	J	K	Q	→	2
D	T				→	3
L					→	4
M	N				→	5
R					→	6
F	V				→	7
S	X	Z			→	8

Untuk setiap huruf dalam basis data dapat diklasifikasikan mcnjadi tiga tingkat, yaitu (1) sama, (2) mirip, dan (3) tidak berhubungan sama sekali. Tingkat mirip dapat dibagi menjadi tiga tingkat yang lebih rendah, yaitu (2a) bunyi akhir mirip, (2b) prefiks bunyi akhir mirip, dan (2c) bunyi akhir berbeda.

Kesamaan String. Metode yang terkenal untuk mcmbandingkan string adalah n-grams. Teknik n-grams ini hanya mcmbandingkan huruf pada kata tanpa mcmpcrhatikan bahasa yang digunakan. Jika dua buah string dibandingkan dengan memperhatikan pada n-gramsnya, maka himpunan n-grams akan dihitung untuk kedua string. Kcmudian kedua himpunan ini dibandingkan dan semakin banyak n-grams yang sama muncul pada kedua himpunan, maka kedua string tersebut semakin mirip.

Pendekatan umum pada perhiyumgan koefisien kesamaan (similarity coeficient [sc]) ini ditunjukkan oleh persamaan (1), dimana N1 dan N2 adalah himpunan n-grams dari dua kata yang akan dibandingkan.

$$sc = \left|\frac{NI \cap N2}{NI \cup N2}\right| \qquad (1)$$

Tabel 3 menunjukan sebuah contoh penggunaan trigram. Trigram adalah himpunan urutan huruf-huruf dalam suatu kata sebanyak tiga buah, contohmya kata RECEIVE mempunyai lima trigram, yaitu REC, ECE, CEI, EIV, dan IVE. Misalkan pengguna melakukan pencarian kata RECEIVE yang salah eja menjadi kata RECEIEVE. Di lain pihak basis data hanya mengandung lima kata yang mirip dengan kata yang dicari, yaitu kata RECEIVE, RECEIVER, REPRIEVE, RETRIEVE, dan REACTIVE. Sekarang himpunan trigram dihitung dan sebagai contoh, dari pembandingan kata RECEIEVE dan RECEIVE, kata RECEIVE mempunyai tiga dari delapan trigram yang berbeda, sehingga kata tersebut mendapat koetisicn kesamaan 3/8, yang cocck dengan kata RECEIVE yang dicari.

Ada dua parameter yang dapat dipilih ketika memakai metode yaitu (1) panjang *n-grams* dan (2) spasi tambahan. Untuk parameter panjang *n-grams*, trigram, dan digram memperoleh hasil yang paling baik dalam pengambilan kata-kata yang mirip pada setiap kata.

Untuk parameter yang mcnggunakan spasi tambahan, analisis *n-grams* dapat menambahkan spasi tambaban pada awal dan akhir kata. Teknik ini menekankan persamaan pada awal dan akhir huruf pada kata. Contohnya kata IDLE dipetakan ada dua trigram IDL dan DLE jika tidak ada spasi tambahan. Sedangkan bila menggtmakan satu spasi tambahan, hasilnya adalah _ID dan LE_ sebagai tambahan bila menggunakan dua spasi tambahan, akan diperoleh trigram _I dan E__ sebagai tambahan.

Ukuran Kesamaan yang digunakan pada n- grams mirip dengan modal ruang vektor untuk temu kembali teks. Pada kasus ini, istilah *n-grams* serta dokumen dan *query* adalah sebuah kata yang diciptakan pada sct *n-grams*nya. Dengan analogi ini, seseorang dapat mcnggunakan struktur akses yang sama. Artinya ada *inverted index* untuk n- grams.
Walaupun demikian sebuah masukan tunggal Seharusnya tidak menunjuk pada sebuah dokumen, karena sebuah dokumen akan mcngandung beberapa kata benda yang tepat (*proper noun*) dan informasi tambahan akan dibutuhkan untuk menghindari kesalahan koordinasi n-grams yang dimiliki kata-kata yang bcrbeda. Walaupun begitu, sebuah masukan *inverted list* seharusnya mereferensikan sebuah masukan pada kamus kata benda yang tepat yang muncul pada basis data. Bila melihat pada panjang rata-rata kata, akan lebih tepat untuk mcnyimpan kata benda yang tepat langsung pada *inverted index* untuk menghindari akses tambahan pada kamus.

Tabel 3. Contoh penggunaan analisis trigram

	REC	ECE	CEI	EIV	IVE	ECE	CEI	EIV	VER	ERR	RRE	REP	EPR	PRI	RIE	IEV	RET	ETR	TRI	RIE	IEV	EAC	ACT	CTI	TIV	VE
RECEIEVE	1	1	1	1	1	1																					6/6
RECEIVE	1	1	1				1	1																			3/8
RECEIVER	1	1	1				1	1	1																		3/9
REPRIEVE				1	1					1	1	1	1														2/10
RETRIEVE				1	1									1	1	1	1										2/10
REACTIVE						1														1	1	1	1				0

Kesamaan yang Bcrhubungan Dengan Kesalahan Pengetikan. Dua metode terkenal untuk melakukan pencarian kata-kata yang berhubungan dengan kesalahan pengetikan adalah metode Damerau-Levenstein-Metric scrta metode Skeleton-Key dan Omission-Key.

Metode Damerau-Levenstein-Metric adalah metode yang menggambarkan perbandingan kata- kata dengan memperhatikan cmpat macam kcsalahan pcngetikan (diberikan contoh untuk kata DAMERAU), yaitu :
1) Penyisipan huruf tambahan, misalnya DAHMERAU
2) Penghapusan sebuah hurutQ misalnya DAMRAU
3) Penggantian sebuah huruf dengan humf lain, misalnya DANERAU
4) Penukaran tcmpat sebuah huruf berurutan , misalnya DAMERUA

Misalkan terdapat dua buah kata yang dinotasikan sebagai s untuk kata kcsatu dan t untuk kata kedua. Metode Damerau-Levenstein-Metric ini dapat menghitung jumlah minimum error dari dua kata tersebut dengan persamaan yang diberikan pada persamaan (2). Variabel i dan j menunjukan posisi huruf-huruf yang dibandingkan pada suatu kata.

$$
\begin{aligned}
&f(0,0) = 0\\
&f(i,j) = \min \{\ f(i-1\ j)+1\\
&\qquad\qquad f(i,\ j-1)+1\\
&\qquad\qquad f(i-1,\ j-1)+d(s_i,t_j)\\
&\qquad\qquad f(i-2,\qquad j-2)+d(s_{i-1},t_j)+d(s_i,t_{j-1})+1\}
\end{aligned}
\tag{2}
$$

Fungsi d pada persamaan (2) adalah fungsi ukuran jarak untuk huruf yang bersifat sederhana dan non-identity. Untuk pengukuran yang lebih teliti dapat dilakukan dengan analisis statistika dari kesalahan pengetikan atau dari posisi geometri *keyboard*.

$$d(si,tj)\begin{cases}0, jika\ si = tj\\1, jika\ si \neq tj\end{cases} \tag{3}$$

Fungsi f(i,j) mcnghitung minimum jumlah error yang membedakan huruf i pertama pada kata pertama dari huruf j pertama pada kata kedua. Maka kedua kata s dan t dengan panjang antara ls dan lt berbedadengan f(ls,lt) error.

Definisi pcngulangan fungsi jarak membuat perhitungan menjadi rumit. Dengan menerapkan aplikasi dari teknik pemrograman dinamis (dynamic programming), kompleksitas dapat ditekan.

Metode Skeleton-Key dan Omission-Key adalah metode-metode yang bersifat tidak tcrlalu kompleks bila dibandingkan dengan metode Damerau-Levenstein-Metric dalam mengatasi kesalahan pengetikan. Walaupun bcgitu metode-metode ini sangat efektif.

Metode Skeleton-Key Berdasarkan ide bahwa konsonan membawa lebih banyak informasi daripada vokal. Skeleton-Key sebuah kata, terdiri dari huruf pertama yang diikuti oleh sisa konsonan dan sisa vokal, keduanya muncul berurutan. Kunci ini mengandung setiap huruf paling banyak satu kali dengan mcnghilangkan duplikasi.

Sedangkau metode Omission-Key bcrdasarkah pada observasi bahwa konsonan dengan frekuensi berbeda dapat dihilangkan selama penulisan. Percobaan menunjukkan bahwa konsonan-konsonan dapat dihilangkan dengan urutan huruf RSTNLCHDPGMFBYWVZXQKJ. Artinya huruf R adalah huruf yang paling sering dihilangkan daripada huruf yang lain. Sedangkan huruf J adalah huruf yang paling jarang dihilangkan daripada huruf yang lain. Omission-Key suatu kata terdiri dari konsonan dengan urutan frekuensi yang tcrbanyak, diikuti dengan huruf vokal dengan urutan kemunculannya. Frekuensi kemunculan setiap huruf pada kunci ini paling banyak adalah satu kali.

Sebagai ukuran kesamaan untuk kedua metode, dilakukan pengurutan kunci secara alfabetik Untuk setiap pasangan kunci, jarak pada setiap urutan adalah kebalikan kesamaannya.

Sebagai jalur akses untuk Skeleton-Key dan Omission-Key, dapat digunakan kunci dengan struktur B-Tree. Untuk kunci pencarian, ditunjuk sebuah posisi kunci ini pada tree. Kunci yang paling mirip adalah yang paling dekat pada posisi ini. Untuk metode Damerau-Levenstein-Metric, implementasi metode ini secara langstmg didasarkan pada pemcriksaan yang lengkap pada seluruh data. Untuk aplikasi yang besar, hal ini menjadi penghalang. Metode penggerombolan (clustering) dapat digunakan untuk menyeleksi bagian-bagian dari data dimana pencarian nama dihubungkan dengan query yang paling mungkin. Tetapi bila dilihat dari ukurannya, ada batasan dalam kemungkinan peningkatan kinerjanya.

3. Algoritme Pendeteksian

Algoritme pengindeksan adalah algoritme yang membentuk struktur data sehingga dapat melakukan péncarian teks dengan cepat. Ada beberapa tipe indeks berdasarkan pendekatan temu-kembalinya, misalnya inverted file, file signature, dan trie.

C. Sistem Evaluasi

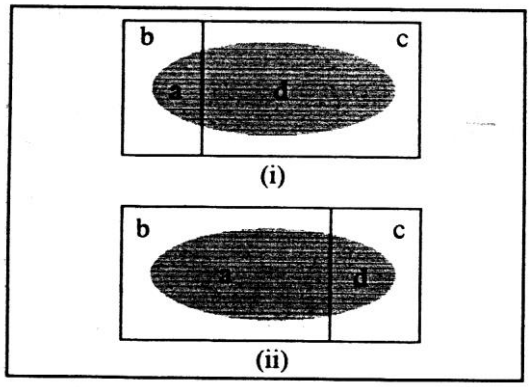
Metode yang digunakan untuk membandingkan teknik-teknik yang berbeda adalah dengan menggunakan metode recall-precision-graph (Salton, 1989). Metode ini telah digunakan sejak lama untuk mengukur efektifitas temu-kembali. Dua parameter yang dipakai pada metode ini adalah recall dan precision. Recall adalah proporsi material relevan yang ditemu-kembalikan yang didefinisikan seperti pada persamaan (4). Sedangkan precision adalah proporsi dari material relevan yang ditemu-kembalikan yang didefinisikan seperti pada persamaan(5).

$$R = \frac{Jumlah\ dokumen\ relevan\ yang\ ditemukan}{jumlah\ dokumen\ relevan\ dalam\ basis\ data} \tag{4}$$

$$P = \frac{Jumlah\ dokumen\ relevan\ yang\ ditemu-kembalikan}{jumlah\ dokumen\ yang\ ditemu-kembalikan} \tag{5}$$

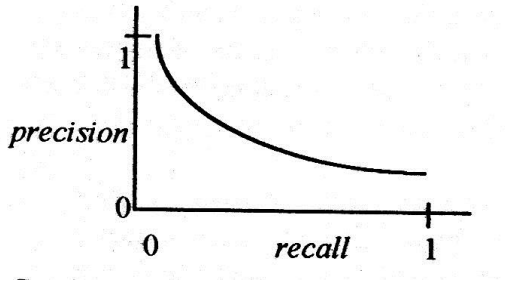
Untuk query-query yang terperinci, nilai parameter precision akan tinggi sebab semua item yang ditemu-kembali relevan. Di pihak lain nilai parameter recall akan rendah karena jumlah item yang ditemu-kembali hanya sedikit. Sedangkan untuk query-query yang tidak terperinci, jumlah item relevan jumlahnya menurun, sedangkan pada waktu yang sama jumlah item tidak relevan yang ditemu-kembali jumlahnya menaik sehingga nilai parameter precision rendah. Jadi pencarian yang terpennci menghasilkan precision yang tinggi dan recall yang rendah, dan sebaliknya .

Situasi tersebut pada Gambar 1, dimana bagian yang ditemu-kembali adalah bagian a dan b. Daerah yang berwarna abu-abu menunjukkan bagian koleksi yang relevan, sedangkan garis vertikal membatasi daerah koleksi yang ditemu-kembalikan dan yang tidak ditemu-kembalikan. Aktivitas penemu-kembalian item-item ini membagi koleksi data menjadi empat bagian (Gambar 1), yaitu (a) bagian koleksi relevan yang ditemu-kembali, (b) bagian koleksi tidak relevan yang ditemu-kembali, (c) bagian koleksi tidak relevan yang tidak ditemu-kembali, dan (d) bagian koleksi relevan yang tidak ditemu- kembali. Terlihat bahwa semakin banyak koleksi relevan yang ditemu-kembalikan, semakin banyak pula koleksi tidak relevan yang ditemu- kembalikan.



Gambar 1. Variasi nilai recall dan precision. (i) Formulasi query yang menghasilkan nilai precision tinggi dan njlai recallrendah dan (ii) formulasi query yang menghasilkan nilai precision rendah dan nilai recall tinggi

Karena kinerja sistem temu-kembalinya sering ingin dibandingkan dengan recall dan precision dalam satu grafiik, maka dikembangkanlah suatu metode. Metode ini menghasilkan plot nilai recall pada sumbu x dan nilai precision pada sumbu y. Telah diketahui bahwa nilai precision dan recall saling berlawanan. Maksudnya ketika nilai precision menaik, nilai recall menurun dan sebaliknya. Gambar 2 mcnerangkan hal-hal tentang grafik recall-precision tersebut. Kinerja rata-rata yang umun terjadi adalah keadaan dimana tingkat recall dan precision antara 0.5 dan 0.6 (Salton, 1989).



Gambar 2. Grafik *recall-precision*

Pada metode recall-precision ini, karena kumpulan jawaban tidak diurutkan secara linear, hanya sejumlah kecil recall-precision-point yang dihasilkan untuk setiap query. Untuk mendapatkan arti dari rata-rata jawaban setiap himpunan query, diperlukan sebuah metode interpolasi baru, yaitu metode probability of relevance. Metode probability of relevance ini mengasumsikan bahwa pengguna mempelajari seluruh peringkat hasil, dimana secara acak diseleksi sebuah dokumen dari peringkat paling atas yang belum dibaca secara lcngkap. Pada tahapan proses ini, precision didefinisikan sebagai peluang bahwa dokumen yang diperiksa secara lengkap adalah relevan. Sedangkan recall adalah bagian dari dokumen-dokumen yang pemakai telah lihat sejauh itu.

Algoritme metode probability of relevance akan ditcrangkan di bawah ini. Misalkan diberikan sebuah peringkat yang mengandung recall item relevan dan i item tidak relevan, dimana s adalah jumlah dokumen relevan yang diambil dari peringkat lf dan r adalah jumlah dokumen relevan dari peringkat lf. Pemakai yang mcnginginkan agar s < r dokumen relevan, akan mcmperoleh nilai esl (expected search length) dokumen yang tidak relevan. Persamaan (6) akan mcmberikan definisi nilai esl.

$$esl_{r,i}(s) = \frac{s.i}{r+1} \qquad (6)$$

Keterangan :
s : jumlah dokumen relevan yang ditemu-kembalikan dari peringkat l_f;
r : jumlah total dokumen relevan dari peringkat l_f
i : jumlah dokumen tidak relevan dari peringkat l_f
f : peringkat yang sedang diamati

Untuk mengatasi situasi dimana ada lebih dari satu peringkat, maka nilai l_f; adalah nilai peringkat yang berada dalam perhatian sehingga persamaan akan berkembang menjadi seperti pada persamaan (7).

$$esl\ (NR) = j + esl_{r,i}(s) = j + \frac{s.i}{r+1} \qquad (7)$$

Kctcrangan :
j : jumlah dokumen tidak relevan dari peringkat 1 sampai l_f,
NR : jumlah dokumen relevan yang diminta

Akhimya dengan mengkombinasikan persamaan (7) dengan definisi *probability of relevance* (PRR), diperoleh persamaan PRR(NR) scpcrti pada persamaan (8).

$$PRR(NR) = \frac{\frac{NR}{NR+esl(NR)}}{\frac{NR}{NR+j+(s.i)/(r+1}} \qquad (8)$$

Jika nilai NR bertipe non-integer, akan diperoleh interpolasi yang halus dari nilai rata-rata jawaban himpunan *query*.

D. Tata Bunyi

Ketika berbicara, alat-alat bicara menghasilkan bunyi. Alat-alat bicara untuk mcnghasilkan bunyi bahasa adalah (1) udara yang dikeluarkan dari paru-paru, (2) artikulator yang merupakan alat ucap yang dapat bergarak atau bergeser, dan (3) titik artikulasi yang merupakan bagian-bagian alat ucap yang mcnjadi tujuan sentuh artikulator (Sudaryat & Natasasmita, 1984). Kerjasama antara ketiga faktor tersebut akan menghasilkan bunyi bahasa.

Penyelidikan bunyi-bunyi yang dipakai dalam tutur itu termasuk bagian ilmu bahasa yang disebut fonologi. Ilmu ini terbagi dua bidang, yaitu:
1) Fonetik, yaitu ilmu yang menyelidiki bunyi bahasa tanpa memperhatikan sebagai pembeda makna
2) Fonemik, yaitu ilmu yang menyelidiki bunyi bahasa dengan memperhatikan fungsinya sebagai pembeda makna.

Dalam fonetik dapat diketahui alat-alat yang dipergunakan untuk menimbulkan bunyi sehingga dapat diketahui pula proses terjadinya bunyi-bunyi itu. Ada tiga jenis fonetik yang perlu diketahui, yaitu:

1) Fonetik organis atau artikulatoris, yaitu cabang ilmu fonetik yang mcmpclajari bagaimana bunyi-bunyi bahasa itu dihasilkan oleh alat ucap manusia.
2) Fonetik akustis, yaitu cabang ilmu fonetik ` yang mempelajari ciri-ciri bunyi bahasa; ilmu interdisipliner dalam linguistik dan fisika. (Krisdalaksana, 1993)
3) Fonetik auditoris, yaitu cabang ilmu fonetik yang mempelajari cara penerimaan bunyi 4 bahasa oleh pendengar.

1. Tata Bunyi dalam Bahasa Inggris

Fonetik Organis. Dalam bahasa Inggris hanya terdapat lima buah huruf vokal, yaitu a, i, u, e, dau o (O'connor, 1973). Suara vokal dalam bahasa Inggris sangat banyak variasinya dari suatu aksen tertentu dengan aksen lainnya, akibatnya terdapat sedikitnya 21 fonem dalam bahasa Inggris, diantaranya /c/ pada kata pet dan /æ/ pada kata pat.

Konsonan dalam bahasa Inggris dapat dibedakan berdasarkan tujuh keistimewaannya, yaitu:

1) *Nasal*,yaitu suara hidung
2) *Fortis*, yaitu suara dengan tekanan yang besar, sedangkan bila tekanannya rendah disebut lenis.
3) *Coronal*, yaitu keadaan ujung lidah yang menaik
4) *Anterior*, yaitu gcrakan lidah didepan langit-langit keras
5) *Continuant*, yaitu hambatan udara
6) *Strident*, yaitu suara desisan
7) *Glide*, yaitu keadaan tengah lidah pada langit-langit keras.

Fonetik Akustis. Suara yang mengalir dari pembicara ke telinga pendengar biasanya merambat melalui medium udara. Suara yang didengar berhubungan erat dengan karakteristik osilasi atau getaran molekul udara, dimana satuan jumlah perputaran molekul udara setiap detik (cycle per second [cps]) discbut frekuensi getaran.
Setiap fonem dalam abjad memiliki pola akustik masing-masing. Karaktcristik ini dipengaruhi oleh frekuensi masing-masing fonem pada suatu selang waktu tertentu .

Fonetik Auditoris. Suara yang didengar oleh pendengar diolah oleh otak untuk menghasilkan pengertian bunyi tersebut (Pike, 1958). Karaktcristik bunyi diantaranya dipcngaruhi oleh kerasnya suara dan kualitas suara.

2. Tata Bunyi dalam Bahasa Indonesia

Huruf-huruf dalam abjad terdiri dari vokal dan konsonan. Vokal adalah bumyi bahasa yang dihasilkan alat ucap manusia apabila udara yang dihembuskan dari paru-paru tidak mcndapat halangan. Huruf-huruf vokal (a, i, u, c, dan o) cenderung sangat sering digunakan dan tidak banyak mengalami perubahan bunyi dalam pengucapannya. Bunyi pengucapan konsonan cenderung tidak mcngalami perubahan bunyi dalam kata.

Suatu huruf dalam abjad disebut konsonan apabila udara yang keluar dari paru-paru ketika pcngucapan mcndapat halangan atau rintangan. Konsonan dapat dibagi ke dalam beberapa kelompok dengan memperhatikan faktor-faktor yang mcnghasilkannya, yaitu :

1) Artikulator dan titik artikulasi
2) Turut tidaknya pita suara bergetar
3) Jalan yang dilalui oleh udara
4) Jenis halangan yang dijumpai tatkala udara keluar

Berdasarkan artikulator dan titik artikulasi, konsonan dibagi menjadi tujuh kelompok, yaitu :

1) Konsonan bilabial (dibentuk oleh kedua bibir), yaitu b, p, m, dan w.
2) Konsonan labiodental (dibentuk oleh bibir dan gigi), yaitu f dan v
3) Konsonan apiko-dental (ujung lidah menycntuh gigi), yaitu t dan n
4) Konsonan apiko-alveolar (ujung lidah menyentuh kaki gigi), yaitu t, d, dan n.
5) Konsonan palatal (tengah lidah menyentuh langit-langit keras), yaitu c dan j
6) Konsonan velar (belakang lidah menyentuh langit-langit lembut), yaitu k, g, dan q
7) Konsonan laringal (pita suara terbuka lebar), yaitu h.

Berdasarkan turut tidaknya pita suara bergetar, konsonan dapat dibagi menjadi dua golongan, yaitu :

1) Konsonan bersuara, yaitu b, d, n, g, w, z, dan sebagainya
2) Konsonan tidak bersuara, yaitu p, t, c, k, f, s, dan sebagainya.

Berdasarkan jalan yang dilalui oleh udara, konsonan dibagi mcnjadi dua kelompok, yaitu :

1) Konsonan oral (bila suara keluar melaluii rongga mulut), yaitu p, b, k, d, w, s, r, dan w
2) Konsonan nasal (bila udara yang keluar melalui rongga hidung), yaitu m dan n.

Berdasarkan jenis halangan yang dijumpai tatkala udara keluar, konsonan dibagi menjadi lima kelompok, yaitu :

1) Konsonan hambat (stop), yaitu p, b, k, t, d, dan lain-lain
2) Konsonan frikatif (bunyi geser), yaitu f dan v
3) Konsonan spiran (desis), yaitu s, x, dan z
4) Konsonan likuida (lateral), yaitul l
5) Konsonan trill (bunyi getar), yaitu r.

Dalam bahasa Indonesia dikenal istilah konsonantal, yaitu sifat aksara yang silabis dan tidak menggambarkan vokal atau keadaan dimana terjadi turunnya frekuensi forman dengan pengurangan intensitas secara menyeluruh yang menandai energi yang rendah. Secara artikulatoris konsonantal menunjukkan hambatan udara terhadap aliran udara di atas glotis (Kridalaksana, 1993). Sifat ini tidak dimiliki oleh huruf vocal dan konsonan h, w, dan y.

III. METODOLOGI

A. Pengambilan Data

Pengambilan data dilakukan setelah penentuan tujuan penelitian dan studi pustaka dilakukan. Dalam penelitian data yang digunakan adalah nama- nama orang (nama depan dan nama akhir) yang umum digtmakan orang Indonesia. Data tersebut diperoleh dari buku petunjuk telepon kota Bogor tahun 1995-1996.

Jumlah data yang dipakai ialah 200 buah. Pengambilan data nama tersebut dilakukan secara acak, kemudian dicari beberapa variannya sampai berjumlah 200 buah. Dalam penelitian ini hanya diolah satu data, yaitu nama depan.

Data nama tcrsebut disimpan dalam tiga buah berkas teks. Data nama depan dan nama akhir tersebut disimpan dalam berkas teks dengan struktur:

NAMA1, string[15]
NAMA2, string[15]

Selain nama depan dan nama akhir, setiap berkas juga memilih data kode, data nilai, dan data peringkat. Data kode adalah data yang berisi kode Phonix yang dihasilkan masing-masing algoritme. Data nilai adalah data sementara hasil pcmbandingan dari pencarian yang dilakukan. Sedangkan data peringkat adalah data dari konversi biner data nilai. Data ini juga merupakan data acuan pengurutan untuk penemu- kembalian infomtasi.

Untuk program dengan metode Phonix4 data kode, data nilai, dan data peringkat disimpan dalam berkas dengan struktur :

KODEP, string[4]
NILAIP, string[5]
TOTALP, byte

Untuk program dengan metode Phonix8, data kode, data nilai, dan data peringkat disimpan dalam berkas dengan struktur :

KODEP, string[8]
NILAIP, string{9]
TOTALP, integer

Sedangkan program dengan metode PhonixE, data kode, data kode bunyi akhir, data nilai, data nilai bunyi akhir, dan data peringkat disimpan dalam berkas dengan struktur :

KODEP1, string{4]
KODEP2, string{4]
NILAIP1, string[5]
NILAIP2, string[5]
TOTALP, integer

B. Pembuatan Program

1) Perangkat Lunak dan Perangkat Keras

Pembuatan program dilakukan dengan menggunakan perangkat lunak Microsoft Access 2002, sedangkan perangkat kerasnya adalah sebuah laptop berprosesor Genuine Intel® T2050 @ 1.6 GHz, 797 MHz, 248 MB RAM dengan *operating system* Windows XP Proffesional.

2) Perancangan Program

Pada penelitian ini dibuat tiga buah program yang melakukan pekerjaan sesuai algoritmenya masing-masing, yaitu :

a. Program Phonix4 untuk mengolah dokumen dengan algoritme Phonix dengan empat huruf
b. Program Phonix8 untuk mengolah dokumen dengan algoritme Phonix dengan delapan huruf
c. Program PhonixE untuk mengolah dokumen dengan algoritme Phonix dengan empat huruf dan bunyi akhir.

Setiap program memiliki 9 buah modul, yaitu (1) modul penambahan *record*, (2) modul penampilan seluruh *record*, (3) modul pencarian *record* berdasarkan nama, (4) modul pencarian *record* berdasarkan nomor dokumen, (5) modul perbaikan *record* berdasarkan nama, (6) modul perbaikan *record* berdasarkan nomor dokumen, (7) modul pembuatan kode *filtering*, (8) modul pencarian dokumen berdasarkan kode *filtering*, dan (9) modul pencetakan berkas basil *query*.

Modul Penambahan *Record*. Modul ini berfungsi untuk menambah *record* pada berkas tertentu yang diaktifkan. Sebagai masukan pada modul ini adalah berkas data yang berisi data nama depan dan nama akhir terdahulu, sedangkan keluarannya adalah berkas yang sama dengan penambahan sebuah dokumen.

Algoritme :
Hitung jumlah *record* yang ada dalam berkas
Baca data *record* baru yang dimasukan.
Tuliskan *record* baru tersebut pada akhir berkas.

Modul Pcnampilan Seluruh *Record*. Modul ini berfungsi untuk seluruh *record* pada berkas tcrtentu. Masukan dan keluaran pada modul ini adalah berkas yang sama

Algoritme :
Untuk seluruh *record* pada berkas, lakukan :
baca *record*
tampilkan *record* pada layar

Modul Pencarian *Record* Berdasarkan Nama. Modul ini berfungsi untuk menampilkan hasil pencarian *record* pada berkas tertentu, dengan asumsi *record* tersebut ada pada berkas yang scdang diaktifkan Pencarian dilakukan Berdasarkan data nama Masukan dan keluaran pada modul ini adalah berkas dengan data yang sama.

Algoritme :
Baca data nama depan dan nama akhir untuk data yang dicari dari monitor.
Untuk setiap *record* pada berkas (pencarian berurutan), lakukan:
Baca data nama depan dan nama akhir.
Bila sama. tampilkan.
Bila tidak ditemukan beri pesan.

Modul Pencarian *Record* Berdasarkan Nomor *Record*. Modul ini berfungsi untuk menampilkan hasil pencarian *record* pada berkas tertentu, dengan asumsi *record* tersebut ada pada berkas yang sedang diaktifkan. Pencarian dilakukan berdasarkan nomor *record*. Masukan dan keluaran pada modul ini adalah berkas dengan data yang sama.

Algoritme :
Baca data nomor *record* untuk data yang dicari dari monitor.
Lakukan pencarian nomor *record* tersebut pada berkas.
Bila ada tampilkan.
Bila tidak ada, beri pesan

Modul Perbaikan *Record* Berdasarkan Nama. Modul ini berfungsi untuk mengoreksi *record* pada berkas tertentu, dengan asumsi *record* tersebut telah ada pada berkas yang sedang diaktifkan. Pencarian *record* berdasarkan data nama. Sebagai masukan pada modul ini adalah berkas yang berisi nama depan dan nama akhir terdahulu, sedangkan keluarannya adalah berkas yang sama dengan sebuah perbaikan pada salah satu *record*-nya.

Algoritme :
Baca data nama depan dan alhir untuk data yang dicari.
Untuk setiap *record* pada berkaslakukan :
Bila ada, lakukan :

Baca nama depan dan akhir pada *record*
Bandingkan dengan data yang dicari
Bila sama, lakukan :
Tampilkan *record*
Baca *record* baru
Tuliskan pada berkas
Bila tidak ditemukan, beri pesan.

Modul Perbaikan *Record* Berdasarkan Nomor Record. Modul ini berfungsi untuk mengoreksi *record* pada berkas tertentu, dengan asumsi *record* tersebut telah ada pada berkas yang sedang diaktifkan. Pencarian *record* berdasarkan data nama. Sebagai masukan pada modul ini adalah berkas yang berisi nama depan dan nama akhir terdahulu, sedangkan keluarannya adalah berkas yang sama dengan sebuah perbaikan pada salah satu *record*-nya.
Algoritme :
Baca data nomor *record* untuk data yang dicari.
Lakukan pencarian *recor* tersebut
Bila ada, lakukan :
Tampilkan *record*
Baca *record* baru
Tuliskan pada berkas
Bila tidak ditemukan, beri pesan.

Modul Pembuatan Kode *Filtering*. Modul ini berfungsi untuk mcmbuat kode sesuai dengan algoritme *filtering*-nya pada berkas tertentu. Sebagai masukan adalah berkas data terdahulu, sedangkan keluarannya adalah berkas yang sama dengan hasil pengkodeannya.
Algoritmenya :
Untuk seluruhnya *record* pada berkas, lakukan:
Baca *record*.
Ubah nama depan menjadi kode algoritma *filtering* masing-masing.
Simpan kode tersebut pada *file* kode.

Modul Pencarian Dokumen Berdasarkan Kode *Filtering*. Modul ini berfungsi melakukan pembandingan nilai kode *filtering query* yang diminta dengan kode *filtering* pada berkas dan mcnampilkan hasil penemu-kembalian informasi tersebut sesuai tingkat kesamaannya. Masukan dan kcluamnnya adalah berkas yang sama.
Algoritme :
Baca *query* pencarian .
Ubah *query* tersebut menjadi kode *filtering* sesuai algoritme *filtering*-nya.
Untuk sebuah *record* pada bakas, lakukan :
Bandingkan kode pada *record* dengan kode *query* pencarian .
Buat data nilai dari nilai pencariannya.
Simpan dalam berkas Urutkan seluruh *record* scsuai nilai peringkatnya dengan algoritme *quick sort*.
Tampilkan seluruh *record*.

Modul Pencarian Berkas Hasil *Query*. Modul ini berfungsi melakukan pencetakan suatu berkas yang diaktifkan melalui alat pencetak (*printer*).
Algoritme : .
Untuk seluruh *record* pada berkas, lakukan :
Baca *record* pada berkas.
Cetak melalui alat pencetak

C. Pengujian Program

Program kemudian diuji dengan menggunakan beberapa data contoh sebelum digunakan untuk data yang sebenamya. Pengujian ini berguna untuk mcnyenyempurnakan program yang telah dibuat. Percobaan setelah seluruh program diuji cara kerjanya, kemudian dilakukan percobaan. Percobaan ini dilakukan dengan mencobakan beberapa *query* yang sama untuk ketiga metode. Untuk setiap *query* telah diketahui kelompok dokumen yang relevan dan tidak relevan yang dicari secara manual. Kelompok dokumen yang relevan sering disebut dengan varian.

Percobaan tersebut dilakukan sebanyak 12 kali. Dalam kedua belas percobaan tersebut, enam percobaan dilakukan dengau *query* yang mcmiliki bunyi akhir (Umar, Iman, Muhammad, Abdul, Abdurrachman, dan Sjamsudin) dan enam kali dilakukan dengan *query* yang tidak memiliki bunyi akhir (Desi, Candra, Tuti, Joko, Dani, dan Budi). Tabel 5 menampilkan keduabelas *query* tersebut, jumlah varian, dan varian-varian setiap *query*.

Secara singkat, percobaan dilakukan dengan beberapa tahap sebagai berikut :

1) Seluruh data masukan yang telah disimpan dalam berkas, misalkan d_j, dengan algoritme *filtering* diubah menjadi kode *filtering*, misalkan dinotasikan dengan d_{ij}, sedangkan i adalah kode jenis metode (i=1 untuk metode Phonix4, i=2 untuk metode Phonix8, dan i=3 untuk metode PhonixE), dan j = 1...200.
2) Keduabelas *query* yang dinotasikan dengan q_k diuji coba dan ditransformasi mcnjadi kode *filtering* q_{ik}, dengan k = 1..12
3) Tandai seluruh data yang dianggap relevan (varian) secara manual.
4) Hitung ukuran kesamaan antara *query* dengan seluruh data, misalkan s_{ij}, sedangkan s_{ij} ialah kesamaan *query* metode ke-i, dokumen ke-j.
5) Hitung nilai peringkat yang diperoleh dengan konversi biner nilai s_{ij}.
6) Urutkan seluruh data secara menurun menurut s_{ij} yang diperoleh.
7) Hitung nilai *recall* dan *precision* scsuai persamaan (4) dan (5) . Nilai tingkat *recall* diperoleh dengan membagi jumlah dokumen yang dianggap relevan menurut persentilnya. Dokumen yang dianggap relevan adalah seluruh dokumen yang memiliki nilai peringkat > 2 untuk metode PhonixE atau nilai peringkat > 1 untuk ketiga metode lainnya. Kemudian pada setiap persentil dihitung proporsi varian terhadap jumlah dokumen yang relevan untuk memperoleh nilai *precision*.

Analisis Data dan Pengambilan Kesimpulan

Pada tahap ini dilakukan analisis pada hasil-hasil pengolahan data yang dikaitkan dengan tujuan penelitian yang diajukan. Analisis dilakukan dengan membentuk tabel *recall- precision*.

Dari nilai-nilai *recall* dan *precision* pada table, kemudian dibentuk grafik *recall-precision* yang menggambarkan kinerja masing-masing metode. Untuk setiap percobaan, nilai *recall* yang diambil adalah dari 0.1,

0.2, 0.3, ..., 1.0. Dari hasil analisis tersebut kemudian diambil suatu kesimpulan yang menjadi hasil penelitian.

IV. HASIL DAN BAHASAN

A. Modifikasi program

Pada prinsipnya, cara kerja Phonix adalah sebagai berikut:

1) Jika huruf pertama dan kedua adalah OE ganti dengan U, SJ dengan S, SY dengan S, TJ dengan C, DJ dengan J dan DH dengan D.
2) Jika huruf kedua seterusnya berturut-turut adalah CH ganti dengan H, SJ dengan S, SY dengan S, TJ dengan C, DJ dengan J.
3) Jika huruf pertama adalah huruf hidup atau konsonan Y, maka ganti huruf pertama tersebut dengan huruf V.
4) Buang bunyi akhir (ending sound) dari kata. Secara kasar bunyi akhir adalah bagian sesudah huruf vokal terakhir atau huruf Y.
5) Buang semua huruf hidup, konsonan H,W, dan Y, dan semua huruf sama yang berurutan.
6) Buat kode Phonix dari kata tersebut tanpa bunyi akhir dengan mengganti semua huruf yang tersisa dengan nilai numerik seperti pada Tabel 2, kecuali huruf pertama. Panjang maksimum kode Phonix dibatasi sampai delapan karakter.
7) Buat kode Phonix dari bunyi akhir dengan mengganti setiap huruf dengan nilai numeric.

Fonem KH tidak perlu diganti dengan H dan fonem DH tidak perlu diganti dengan D, karena algoritme di atas secara otomatis akan melakukan pengkodean yang menghasilkan bunyi yang sesuai

B. Karakteristik dokumen

Ketiga berkas yang dipergunakan pada percobaan, masing-masing terdiri dari 200 dokumen. Dalam dokumen-dokumen tersebut terdapat sejumlah dokumen yang memiliki data nama depan dan nama akhir yang masih menggunakan ejaan Van Ophuysen dan Suwandi, belum menggunakan Ejaan Yang Disempurnakan (EYD). Contohnya adalah penggunaan fonem ‘oe’ pada kata ‘Oemar’, fonem ‘tj’ pada kata Tjipto, dan fonem ‘dj’ pada kata ‘Djoko’. Ketidakseragaman ejaan nama ini yang akan memperlihatkan apakah terjadi perbaikan hasil pada proses penemuan-kembalian informasi. Selain ejaan yag tidak seragam, terlihat pula bahwa dokumen-dokumen dalam berkas pun tidak seluruhnya memiliki nama akhir.

C. Hasil percobaan

Pada percobaan diperoleh empat kode *filtering*, yaitu :

1) Kode Phonix4 yang terdiri dari empat karakter, tanpa kode bunyi akhir.
2) Kode Phonix8 yang terdiri dari delapan karakter, tanpa kode bunyi akhir.
3) Kode PhonixE yang terdiri dari empat karakter, dengan empat huruf kode bunyi akhir.

Setelah kode-kode *filtering* terbentuk, maka dilakukan 12 percobaan dengan *query-query* yang berbeda. Lampiran 2 sampai dengan 5 menampilkan hasil perhitungan *recall-precision* dari ketiga metode, yaitu Phonix4, Phonix8 dan PhonixE. Nilai rata-rata *recall-precision* masing-masing metode dicantumkan pada tabel 6. Untuk memperjelas perbedaan hasil dari keempat metode tersebut, pada gambar 3 ditampilkan grafik ketiga metode tersebut secara bersamaan.

Seluruh grafik pada gambar 3 secara jelas menampilkan perbedaan kinerja setiap metode. Dari grafik tersebut dapat diketahui bahwa metode Phonix8 menghasilkan kinerja paling buruk. *Precision* metode ini pada tingkat *recall* 0.1, memiliki nilai ± 0.08 lebih rendah daripada metode Phonix4 dan ± 0.05 lebih rendah drapada metode PhonixE

Nilai *precision* metode Phonix4 pada tingkat *recall* 0.1 lebih baik daripada metode PhonixE. Nilai *precision* pada tingkat *recall* pada selang 0.2-0.3 sangat dekat di bawah metode PhonixE. Sedangkan selang tingkat *recall* 0.3-1.0 sangat dekat kinerjanya dengan PhonixE.

Metode PhonixE secara umum memberikan kinerja tdak jauh berbeda bila dibandingkan dengan kedua metode lainya pada selang tingkat *recall* 0.1-0.3. Pada tingkat *recall* selanjutnya, metode ini memiliki nilai *precision* yang sangat terbaik di antara ketiga ametode.

D. Analisis Nilai Recall –Precision

Pada sebab ini akan dianalisis kinerja masing-masing metode secara berurutan. Dari yang berkinerja terburuk sampai dengan yang terbaik. Urutan tersebut yaitu metode Phonix8, Phonix4 dan PhonixE.

a. Phonix8 yang dimodifikasi

Walaupun metode Phonix8 menggunakan delapan kelompok konsonan, tetapi memiliki kinerja kedua terburuk hal ini disebabkan oleh terlalu panjangnya kode *filtering* yang dihasilkan. Dengan semakin panjangnya kode, semakin banyak pula kode yang memiliki karakater yang sama pada posisi tersebut. Akibatnya jumlah dokumen yang relevan yang ditemu-kembalikan akan terlalu banyak.

Hal tersebut lebih terlihat lagi pada *query* yang cukup panjang. Sebagai contoh *query* Abdurrachman dengan menggunakan metode Phonix8 pda tingkat *recall* 0.1 memiliki nilai *precision* 0.5. sedangkan metode Phonix4 dan PhonixE memliki *precision* lebih tinggi lagi yaitu 1.

b. Phonix4 yang dimodifikasi

Metode Phonix4 telah menggunakan delapan kelompok konsonan dan telah memperpendek panjang kode yang dihasilkan sampai empat huruf untuk menekan jumlah dokumen relevan yang ditemu-kembalikan. Hal ini sebenarnya sangat baik, hanya saja pada *query* yang cukup panjang hal ini menjadi bumerang.

c. PhonixE

Metode PhonixE selain menggunakan metode Phonix4, juga menggunakan empat karakter tambahan nilai bunyi akhir untuk memberikan tambahan nilai tingkat kesamaan. Apabila ditelaah lebih lanjut untuk *query* yang cukup panjang dan memiliki bunyi akhir, nilai *precision* pada suatu tingkat *recall* yang lebih tinggi daripada kedua metode lainnya. Sebagai contoh *query* Abdurachman. Lebih lanjut dapat dikatakan bahwa metode ini lebih baik dibandingkan

dengan metode Phonix4 karena penurunan nilai *precision* pada setiap tingkat *recall* tidak terlalu drastis.

E. Tingkat Recall yang Optimal

Dalam berbagai keadaan, kinerja rata-rata yang umum terjadi adalah keadaan dimana tingkat *recall* dan tingkat *precision* antara 0.5 – 0.6. Ketiga metode ini telah mencapai precision antara 0.5-0.6 pada tingkat recall 0.1-0.2, kecuali metode Phonix4 mengalami penurunan precision pada tingkat recall 0.2. Maka pemilihan metode yang akan digunakan sebenarnya yang paling cocok adalah menggunakan metode PhonixE. Pemilihan tersebut bergantung kembali dari isi masing-masing basis data, yaitu banyak sedikitnya data nama yang memiliki bunyi akhir. Apabila banyak data nama yang tidak memiliki bunyi akhir, maka metode Phonix4 yang digunakan sebaliknya apabila banyak data nama yang memiliki bunyi akhir, maka metode PhonixE yang digunakan.

Sebagai catatan, pada hasil percobaan tidak selalu dokumen yang ditemu-kembalikan dari *query* yang diminta berada pada urutan pertama. Hal ini karena terdapat jumlah dokumen yang memiliki kode *filtering* yang sama, sehingga *field* nilai dan *field* total memiliki nilai yang sama pula. Akibatnya pada proses pengurutan, dokumen yang dicari tidak selalu berada pada urutan pertam.

6. Analisis Algoritme

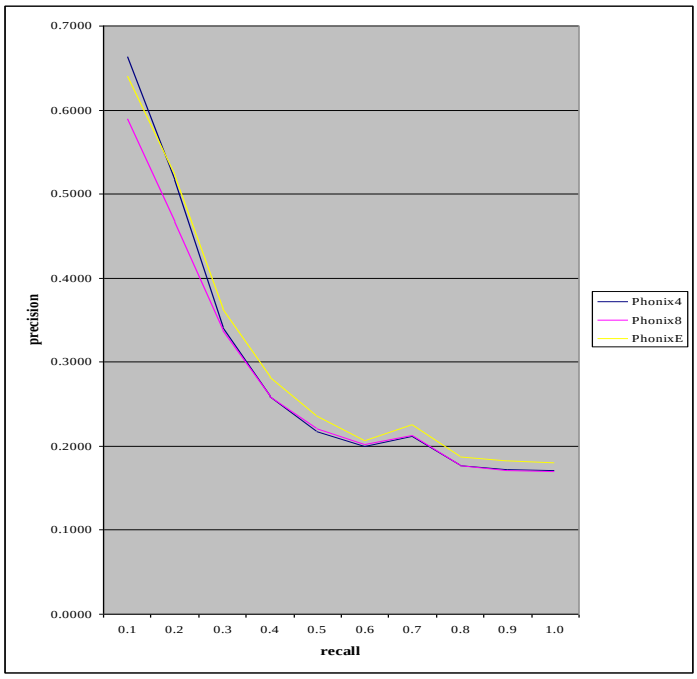
Modifikasi algoritme ini ternyata cukup baik bila dibandingkan hasilnya dengan hasil penelitian awal yang dilakukan. Pada awal penelitian kata CANDRA dan TJANDRA memiliki kode yang berbeda, yaitu C536 dan T253. Dengan adanya modifikasi algoritme kata TJANDRA memiliki kode yang sama, yaitu C536. Akibatnya nilai *precision* menaik pada tingkat *recall* yang sama.

Pengelompokan konsonan-konsonan dalam algoritme Phonix cenderung dilakukan dengan hanya memperhatikan tiga faktor keistimewaan konsonan pada jenis fonetik organis, yaitu (1) nasal, (2) anterior dan (3) strident. Sedangkan penggabungan huruf b dan p dengan huruf f dan v dan penggabungan huruf c, g, j, k dan q dengan huruf s, x dan z dapat dipastikan karena memperhatikan faktor-faktor konsonan pada fonetik akustik. Hal ini sangat mirip dengan penggolongan konsonan bahasa Indonesia.

Dalam pembentukan kode Phonix huruf-huruf pertama dalam suatu kata dibiarkan tetap dan tidak berubah. Hal ini karena huruf pertama dianggap membawa informasi yang lebih banyak. Bila terjadi perubahan huruf vokal dan huruf y dengan huruf v pada algoritme Phonix, hal ini pada algoritme Phonix, hal ini karena huruf vokal sering mengalami perubahan bunyi. Sedangkan penghapusan yang terjadi pada sebuah huruf ganda kan mempengaruhi dan mempersulit proses penemu-kembalian suatu informasi.

Sedangkan untuk kekurangannya, metode ini masih memiliki kelemahan,yaitu

1) Ejaan fonem j pada ejaan lama, seperti pada kata JULI seringkali sebenarnya dibaca YULI. Algoritme ini tidak mampu membedakannya.
2) Masih banyaknya kata yang dikodekan sama karena pembentukan kode yang tidak memperhatikan huruf vokal. Contohnya adalah pada pencarian kata IIN yang mempunyai kode V. Ikut ditemu-kembalikan pula kata-kata selain variannya seperti YAYUK dan YAYAT yang sama-sama mempunyai kode V. Akibatnya kinerja suatu metode menurun karena jumlah dokumen yang relevan akan meningkat dan nilai *precision* yang menurun.



Gambar 3. Perbedaan Hasil dari Keempat Metode

V. PENUTUP

Algoritme Phonix yang dimodifikasi memiliki kinerja yang lebih baik daripada algoritme Phonix yang asli. Algoritmen PhonixE paling baik kinerjanya diantara ketiga varian Phonix. Sedangkan kinerja yang lebih baik antara metode Phonix4 dan Phonix8 pada suatu basis data, sangat ditentukan oleh karakteristik dokumen dalam basis data itu sendiri.

Algoritme Phonix yang menggunakan kesamaan fonetik dapat bekerja dengan baik pada basis data yang berisi dokumen-dokumen berbahasa Indonesia. Hal ini karena konsonan-konsonan dalam bahasa Indonesia dapat digolongkan (dengan keistimewaan-keistimewaan yang sama) menjadi golongan-golongan yang mirip sekali seperti pada Phonix. Tingkat recall yang cukup optimal pada sistem temu-kembali dengan metode Phonix adalah 0.1.

VI. DAFTAR PUSTAKA

Adisantoso, J. 1997. Temu-Kembali Informasi Menggunakan Peluang Bersyarat. Tesis. Program Studi Ilmu Komputer, Universitas Indonesia, Jakarta.

Frakes, W. B. & Baeza-Yates. 1992. Information Retrieval : Data Structures & Algorithm. Prentice-Hall, New Jersey.

Kridalaksana, H. 1993. Kamus Linguistik. Ed ke-3. PT Gramedia Pustaka Utama, Jakarta

O'Connor, J.D. 1973. Phonetics : a Simple and Practical Introduction to Nature and Use of Sound in Language. Penguin Books, England.

Pfeifer, U., T. Poersch & N. Fuhr. 1996. Retrieval Effectiveness of Proper name Search Methods. Information Processing & Management 32 : 667-679.

Pike, K. L., 1958. Phonetics : a Critical Analysis of Phonetics Theory and a Technic for the Practical Description of Sounds. Ed. ke-6 Ann Arbor, Michigan.

Salton, G. 1989. Automatic Text Processing : the Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley Publishing Company, Canada.

Sudaryat, N & H. Natasasmita. 1984. Ringkasan Bahasa dan Sastra Indonesia. Ganeca Exact, Bandung.